Knowledge Management in Optical Networks

Luis Velasco*, Fatemehsadat Tabatabaeimehr, and Marc Ruiz
Optical Communications Group (GCO), Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
**e-mail: lvelasco@ac.upc.edu*

ABSTRACT
Autonomous network operation realized by means of control loops, where prediction from machine learning (ML) models is used as input to proactively reconfigure individual optical devices or the whole optical network, has been recently proposed to minimize human intervention. A general issue in this approach is the limited accuracy of ML models due to the lack of real data for training the models. Although the training dataset can be complemented with data from lab experiments and simulation, it is probable that once in operation, events not considered during the training phase appear thus leading into model inaccuracies. A feasible solution is to implement self-learning approaches, where model inaccuracies are used to re-train the models in the field and to spread such data for training models being used for devices of the same type in other nodes in the network. In this paper, we develop the concept of collective self-learning aiming at improving models error convergence time, as well as at minimizing the amount of data being shared and stored. To this end, we propose a knowledge management (KM) process and an architecture to support it.
Keywords: Knowledge Management; Network Automation; Autonomic Transmission; Self-learning

1 INTRODUCTION

The optical network is being extended toward the edges of operators' networks [1], fostered not only by the increased amount of traffic coming from current and future access segment, but also by the stringent requirements that they need to support, like low latency and high reliability. The added complexity, in addition to highly dynamic traffic, requires the network operation to be automated. In this regard, autonomous *control loops* based on Machine Learning (ML) techniques [2] have been proposed aiming at reducing human intervention as a way to minimize network operational costs. In general, an autonomous control loop *uses knowledge* discovered during a ML training phase to predict (near) future network conditions, so as to proactively prepare resources to deal with them (*decision-making*).

In view that knowledge usage and decision making are needed not only at the Software-defined Networking (SDN) controller level, but also at the local node/subsystem level, the control plane should be designed to support such variety of use cases and scenarios of autonomous networking. For instance, the authors in [3] present the benefits of adding a Monitoring and Data Analytics (MDA) system and present operators use cases looking at automating optical network operation.

Enough real data to produce accurate ML models is rarely available owing to a plethora of reasons, like the existing legal and regulatory context that limits the availability of real network performance measurement, as well as the difficulty to obtain training datasets belonging to specific pre-commercial and commercial technologies and use them in current and forecasted scenarios. In view of that, the authors in [4] proposed a learning life-cycle to facilitate ML deployment in real operator networks. In particular, they added a ML training phase to be carried out after detecting *model inaccuracies* (e.g., in the form of prediction errors), being this the basis of *self-learning* to progressively improve the ML models deployed in the network. Such improvement can be made faster in the case of the model is being used by several agents, which can share model's inaccuracies among them; they called this as *collective self-learning*. It was demonstrated that collective self-learning outperforms individual strategies. However, because the size of the training dataset might be large to reach high-accuracy and robustness, (*data-based*) collective self-learning increases data to be stored and to be exchanged among agents.

Instead of data, ML models can also be shared among agents. An example of such model sharing can be found in [5], where the authors proposed to model OD traffic in the core as an aggregation model of the conveyed metro flows models. In this case, metro flow models are trained by the metro SDN controllers and shared with the core SDN controller, which composes the model for the core OD.

In this paper, we go further and target at completing the knowledge management (KM) process for truly autonomous optical network operation. The KM process entails creating and sharing knowledge and it has been applied to achieve organizational objectives, like continuous improvement of an organization. Those *learning organizations* are able to adapt quickly and effectively to be superior to the competitors in their field or market [6]. Here, we apply KM in the context of optical transmission and networking and define it as the process to autonomously (i.e., without human intervention) *i*) *discover*; *ii*) *share*; *iii*) *assimilate*; and *iv*) *use* knowledge to improve the performance of a network. Note that networks, like organizations, consist of a set of networking devices, which would probably not achieve a global improvement in case of knowledge being individually managed.

2 KNOWLEDGE MANAGEMENT

Fig. 1 presents the architecture proposed to enable KM, where two software agents in charge of networking devices are represented. Agents collect monitoring/telemetry data from the underlying device(s) e.g., an optical transponder (step 1 in Fig. 1a) that are consumed by a ML-based application, to produce some output (e.g., prediction) based on some ML models regarding some device/entity, e.g., the QoT of an optical connection. The results can be used by a decision maker module (2) to tune configuration parameters in the device(s) (3). Note that we just described the typical control loop (1-2-3), which focuses exclusively on *knowledge usage*.

Now let us assume that the output produced by the ML-based application based on the measured data is stored (4) and that such output could be compared to real data measured from the device(s) after some time. If this would be possible, we could conceive an algorithm that would monitor the accuracy of the current ML models and detect events for which the models return inaccurate output (5). For illustrative purposes, Fig. 2a shows an example where a model for regression has been trained with data points. Note that those data points do not need to be uniformly distributed in the regions and can form data clusters in some regions of the *features space*, whereas no data points can be found in other regions. A prediction for data in an unknown region would produce a response value that might be far from the actual response measured from the network. Thus, detecting such inaccuracies would open the opportunity to increase our training dataset with new labelled data (i.e., <*X*, *y*>, where *X* is the input data and *y* the predicted response) and apply ML training to produce more accurate ML models that can be immediately used by the ML-based application (6). This loop (4-5-6) entails *knowledge discovery* and it is the base for *self-learning* [4].

As an alternative to the single ML model covering the complete features space, one could analyze the structure of the training dataset and realize of the presence of data clusters. In such case, specific and more accurate ML models could be produced within each of the selected regions as it is suggested in the example in Fig. 2b (regions R1..3). In this case, some information (*meta-data*) is needed to specify the region of applicability of the model, as well as other important data, like the number of samples used to produce the model, etc. In addition, note that the lack of a model in the region of a collected measurement reveals a new unknown region; those collected data need to be stored until the corresponding label is obtained and can be used to extend the knowledge to that region.

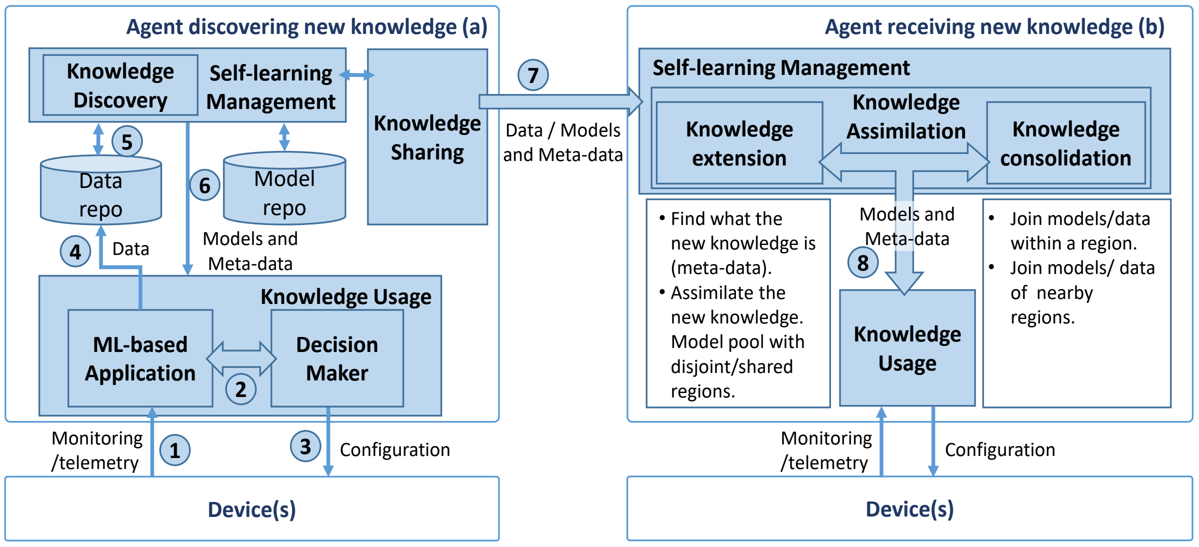

Fig. 1. KM Process. New knowledge is discovered (a) and assimilated for operation (b).

Imagine now that the knowledge discovery process is performed individually per every different device/entity, as the measured data could be specific for such device/entity and so the corresponding ML models. In such case, knowledge discovered from one device/entity cannot be shared among different devices/entities. However, let us assume that either the measured data can be used unchanged by other devices/entities or there exists a function that normalizes the measured data (i.e., removes local dependences) so that the resulting normalized data can be used to train ML models for other devices/entities.

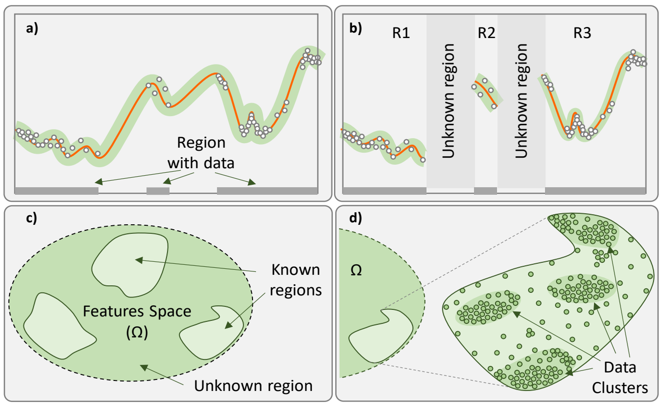

Fig. 2. Known and unknown regions in the features space.

Then, new *knowledge* in the form of labeled data can be *shared* with other agents as soon as it is discovered (7), thus enabling *collective learning* [4]. Note that the normalized data received from other agents can be used to complement the local training dataset; this increases the learning speed since the probability of rare events to be observed increases as there are more observers. However, sharing knowledge in the form of labeled data might entail the exchange of large volumes until the accuracy of the ML models does not reach high values. Note that one single labeled data point consists of a tuple of values and that a complete training dataset can contain a large amount of data points. Another alternative to reduce the amount of data being exchanged is to produce specific models for the knowledge just discovered. These models can be very accurate in a particular region of the features space where the new knowledge has been discovered.

The components related to KM in the agent receiving the new knowledge are sketched in Fig. 1b. Note that the separation between the agent receiving the new knowledge and the one discovering it is done for illustrative purposes, as there is no limitation about being actually the same agent. When a model and meta-data are used to share new knowledge, the receiving agent needs to *assimilate* such knowledge, starting by understanding what the new knowledge is. Assuming that the feature space is modeled in a per-region way, the received knowledge can be located (totally or partially) in one or more of the known regions or in the unknown region; in the former, the model is added to the found region(s) and a merge of regions could be performed, whereas in the latter, a new region is created. We name *knowledge extension* to the process of identifying the new knowledge and updating the regions. Note that a region can be modelled using one or more models, so region updating would entail generating a new model joining the previous model with the received one, or just adding the new model to the pool of models. Another process that we call *knowledge consolidation* is in charge of joining models within a region and joining nearby regions. Fig. 2c-d illustrate the features space of a given problem, where the training dataset contains labeled data grouped into three different regions. However, data points are not usually uniformly distributed along a region, as regions are dynamically re-defined as a result of a region merging process, triggered whenever new knowledge arrives.

Finally, changes in the regions and models and meta-data generate new *operational* models that are ready for knowledge usage (step 8 in Fig. 1b).

3 PROPOSED ARCHITECTURE

Fig. 3 presents an extended architecture for KM, where more details of the agent are depicted; specifically, *knowledge discovery and knowledge assimilation* in the form of extension and consolidation (collectively named *self-learning*), *knowledge sharing*, and *knowledge usage* components are detailed. In addition, the *Knowledge Manager* component coordinates KM operations.

The data collected from the underlying physical device(s) is processed by an *application manager* that uses knowledge for the autonomous control of the device(s). For the sake of generalization, we consider that the configuration of the devices is based on a set of algorithms for different problems, which generate outputs to a decision maker module in charge of finding the best configuration for the forecasted conditions.

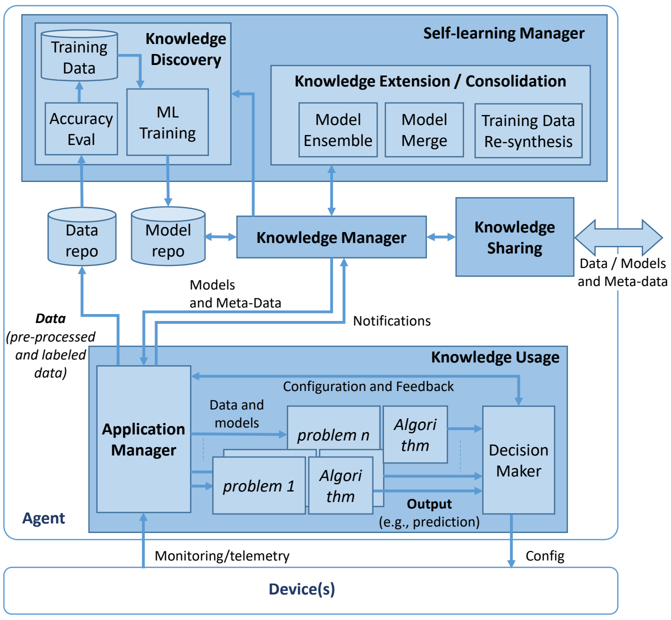

Fig. 3. Detailed architecture for KM

Any problem might require a specific procedure combining several techniques (ML, statistics or mathematics) to generate its outputs. The role of the application manager in the device control loop is to feed the different problems with the required inputs and to adjust the decision maker according to the observed local performance.

In addition to these operational tasks, the application manager exports pre-processed and labeled data (including model predictions and real measurements) to be stored in the *data repository*. Such data is analyzed by the knowledge discovery module, which holds two essential roles: *i*) to identify inaccuracies in the current ML models and, *ii*) to populate its internal training dataset and perform ML training to produce new models that are stored in the *model repository*. The knowledge discovery loop is the main source of knowledge acquisition coming from real data from the operation of the underlying device(s). Such new knowledge can be afterwards shared with other agents through the knowledge sharing module thus, implementing collective self-learning. Consequently, knowledge discovered by other agents is also received and stored in the model repository.

The activity of knowledge discovery could lead to many ML models being stored in the repository, which would hinder knowledge usage. For example, in the case of keeping several ML models restricted to narrow region in the feature space or alternatives models for the same region. Owing to that fact, knowledge assimilation applies methods for knowledge extension and consolidation focused on reducing the number of models used for operation while keeping its overall accuracy. As illustrated in Fig. 3, we consider three different methods for such task, named *model ensemble*, *model merge*, and *training data re-synthesis*. The next section is devoted to providing the details for these assimilation methods.

Finally, following a given scheduling policy, e.g., every time a new ML model is made available or with some periodicity, the knowledge manager updates the ML models of every problem in the knowledge usage module, so the algorithms can use them for operational purposes.

Last but not least, the knowledge usage module plays a pro-active role to speed-up knowledge discovery, as the algorithm can discover that some given measured data locates into an unknown region of the features space of their problems. In such case, the application manager notifies the knowledge manager, which requests the knowledge sharing module to ask other agents about labeled data around the measured one, so as to produce a specific ML model for that unknown region.

4 SUMMARY

The Knowledge Management (KM) process has been proposed aiming at a truly autonomous optical network operation. KM is based on four main pillars: *i*) knowledge discover; *ii*) knowledge share; *iii*) knowledge assimilate; and *iv*) knowledge usage. These pillars allow optical networks to autonomously discover and disseminate knowledge that can be used to adapt its configuration to variable conditions without human intervention.

A general architecture to support KM has been proposed that extend beyond typical control loop implementation and allows for knowledge sharing among different agents disregarding they run distributed in the network nodes or centralized in a controller, like the Monitoring and Data Analytics (MDA) one. Such knowledge sharing enables collective self-learning, which has been demonstrated to reduce models error convergence time.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Commission through the METRO-HAUL project (G.A. nº 761727), from the Spanish MINECO TWINS project (TEC2017-90097-R), and from the Catalan Institution for Research and Advanced Studies (ICREA).

REFERENCES

[1] L. Velasco, P. Wright, A. Lord, and G. Junyent, "Saving CAPEX by Extending Flexgrid-based Core Optical Networks towards the Edges," IEEE/OSA Journal of Optical Communications and Networking, vol. 5, pp. A171-A183, 2013.

[2] D. Rafique and L. Velasco, "Machine Learning for Optical Network Automation: Overview, Architecture and Applications," IEEE/OSA Journal of Optical Communications and Networking, vol. 10, pp. D126-D143, 2018.

[3] L. Velasco, A. Chiadò Piat, O. González, A. Lord, A. Napoli, P. Layec, D. Rafique, A. D'Errico, D. King, M. Ruiz, F. Cugini, and R. Casellas, "Monitoring and Data Analytics for Optical Networking: Benefits, Architectures, and Use Cases," IEEE Network Magazine, vol. 33, pp. 100-108, 2019.

[4] L. Velasco, B. Shariati, F. Boitier, P. Layec, and M. Ruiz, "A Learning Life-Cycle to Speed-up Autonomic Optical Transmission and Networking Adoption," IEEE/OSA Journal of Optical Communications and Networking, vol. 11, pp. 226-237, 2019.

[5] F. Morales, Ll. Gifre, F. Paolucci, M. Ruiz, F. Cugini, P. Castoldi, and L. Velasco, "Dynamic Core VNT Adaptability based on Predictive Metro-Flow Traffic Models," IEEE/OSA Journal of Optical Communications and Networking (JOCN), vol. 9, pp. 1202-1211, 2017.

[6] P. Senge, *The Fifth Discipline: The Art and Practice of the Learning Organization*, Doubleday/Currency, 1990.